~~before runtime, examining a runtime version of a screen element of a graphical user interface to detect an ability to process an input device's events;~~

H2 ~~automatically identifying said screen element as supporting said input device when input device-handling program code is associated with said screen element; and~~

~~automatically examining a superclass class definition of said screen element's class definition if said class definition of said screen element is not identified as supporting said input device.~~

REMARKS

The Examiner is thanked for his review of this application. Claims 10, 15, 18, and 21 have been amended. Claims 1-22 remain pending after entry of the present amendment.

Rejections under 35 U.S.C. § 103

Claims 10 and 12-21 were rejected under 35 U.S.C. 103(a) as being unpatentable over Crutcher et al. ("Crutcher") (U.S. Patent No. 5,844,560) in view of Carey et al. ("Carey") (U.S. Patent No. 6,122,627). These rejections are respectfully traversed.

To establish a *prima facie* case of obviousness, there must be some suggestion or motivation, either in the references or in the knowledge generally available to one having ordinary skill in the art, to combine the references. Additionally, the references when combined must teach or suggest all the claim limitations. As discussed below, the Office has not established a *prima facie* case of obviousness because there is neither suggestion nor motivation, in either the references or in the knowledge of one having ordinary skill in the art at the time of the invention, to have combined the references in the manner proposed. Furthermore, the references when combined do not teach or suggest all of the claim limitations.

Summary of Telephone Interview with Examiner

A telephone interview was conducted between the Applicant's patent attorney Albert S. Penilla (Reg. No. 39,487) and the Examiner Steven Sax on July 23, 2002. The purpose of the interview was to discuss the merits of the case in view of the Examiner's comments contained in the Advisory Action mailed on July 12, 2002. Specifically, in Item 4 of the subject Advisory Action the Examiner stated the following with respect to the Applicant's previous response submitted on June 21, 2002: "Applicant's comments have merit, but the features discussed are not brought out in the rejected claims." The Applicants pointed out to the Examiner that the features discussed were directly cited in the rejected claims.

The discussion in the interview focused primarily on rejected claim 10 out of the set of rejected claims 10 and 12-21. The arguments previously set forth in the Applicant's response of June 21, 2002, were reviewed with the Examiner to ensure clarity of understanding. It was determined that the Examiner's difficulty with the rejected claims resides primarily in the interpretation of the operative term "examining". With respect to claim 10, the context of the disputed term is "... <u>examining</u> a class definition of a screen element of a graphical user interface to detect an ability to process an input device's events; ..."

The Examiner noted that the term "<u>examining</u>" could be interpreted in one instance as "<u>executing</u>". The Applicants respectfully disagreed with the Examiner's interpretation for a number of reasons. Firstly, the Applicants noted that a "class definition" is simply not "<u>executable</u>." A "class definition", as commonly known in the art, is a construction of source code used in an object oriented programming paradigm. The "class definition" must be compiled to a lower level language <u>before</u> it can be executed. For example, in C++ the "class definition" must be compiled to machine language that can be understood by the processor prior to its execution. As another example, in Java the "class definition" must be compiled to bytecode that can be interpreted by an intermediate program such as a web browser. Secondly, the Applicants also pointed out that "examining ... to detect an ability to process an input device's events" is implicitly intended to be performed prior to and without actually processing the input device's events.

Crutcher's Teachings

Crutcher teaches a graphical user interface (GUI) element (e.g., control button) that has a three-dimensional appearance. The GUI element is capable of performing multiple operations each of which is initiated by activation of a particular region of the GUI element. Activation of a region of the GUI element is performed by controlling a positioning device (e.g., mouse) to locate and activate a positioning element (e.g., mouse cursor) over the region of the GUI element. Activation of the positioning element when located over the GUI element causes the GUI element to generate an appropriate signal. Crutcher teaches a means for monitoring the active status and location of the positioning element to determine whether a region of the GUI element is being activated by the positioning device. As a reaction to activation by the positioning device via the positioning element, the appearance of the GUI element is altered to provide visual feedback that the positioning device action has been performed.

Traversal of Rejections based on Crutcher's Teachings

With respect to claim 10, Crutcher does not teach a method of determining input device support of a screen element of a GUI. Rather, Crutcher teaches a response of a GUI element to an input after the input is received from an input device. Furthermore, with respect to claim 10, Crutcher does not teach examining a class definition of a screen element of a GUI to detect an ability to process an input device's events. Crutcher teaches a specific method by which a GUI element processes an input device's event after the input event has occurred. Claim 10 as amended specifically requires that examination of the class definition be performed without executing the class definition of the screen element.

Consider the case in which a user is attempting to activate the GUI element using a mouse as the input device, but the GUI element by design does not support a mouse input device. Without the benefit of the claimed invention, the GUI element will appear normally to the user but will not react to activation by the mouse cursor. Since, the teachings of Crutcher only describe the response of the GUI element after activation by the mouse cursor the user may think that either the program (i.e., GUI element) or the mouse is malfunctioning. However, the reality is that both the program and the mouse are functioning properly. With the claimed invention, the lack of mouse input device support by the GUI element is automatically determined by the computer system and conveyed to the user through visual feedback. Therefore, the user understands prior to attempting activation of the GUI element by the mouse cursor that the GUI element does not support the mouse input device.

With respect to claims 15, 18, and 21, the Examiner asserts the following: "Note that the claim (i.e., claim 1 of Crutcher) is broad and recites that the runtime version of the element is examined and subsequently identified as supporting the input device. This is status indication of the input device, and the element is marked or modified accordingly." The Applicants respectfully disagree with this assertion. The claim does not recite that the runtime version of a GUI element is examined. Rather, the claim of Crutcher recites "... a means for monitoring the active status and location of the positioning element to determine whether a region of the control button is being pushed by the positioning device ..." Per the embodiments of Crutcher, the positioning element refers to something analogous to a mouse cursor, not a GUI element. The control button is analogous to a GUI element. The claim recites monitoring of the positioning element not monitoring of the control button. Therefore, the claim of Crutcher does not recite that a runtime version of the element (i.e., GUI element) is examined. Furthermore, the assertion that the runtime GUI element is "subsequently identified" as supporting the input device may be correct for Crutcher, but teaches away from the claimed invention as discussed below.

The phrase "subsequently identified" refers to the fact that the user must attempt to activate the GUI element with the positioning element via the input device before any knowledge of input device support is obtained. In a manner similar to claim 10, claims 15, 18, and 21 as amended specifically require that examination of the runtime version of the screen element of the graphical user interface be performed without executing the runtime version of the screen element. Thus, the claimed invention does not rely on GUI element activation (i.e., execution) to facilitate observation of the GUI element response in an attempt to identify supported input devices. Rather, the claimed invention considers the input device-handling program code associated with the GUI element to determine if an input device is supported by the GUI element. It is respectfully submitted that the user-implemented trial-and-error method for determining if a GUI element is capable of responding to various input devices, as inferred by the Examiner with reference to Crutcher, is not in any way similar to the apparatuses and methods of the claimed invention. Furthermore, contrary to the claimed invention, such a user-implemented trial-and-error method is a reactive method that is not implementable in a computer program.

Carey's Teachings

Carey teaches development of a tool to be used to obtain different specific "views" of data in a relational database management system. Upon receipt of a query referencing a view type, a query engine generates a query plan that builds mock, or proxy, application type objects in computer memory that do not contain data. An application can run methods on the application objects. The proxy objects can redo the same calculations that a view would do to obtain data. In this way, Carey builds objects from view definitions and queries.

Traversal of Rejections based Carey's Teachings

The Examiner asserts that Carey shows details of class definitions being associated with GUI elements. It is respectfully submitted that nothing in Carey as referenced by the Examiner has anything to do with details of class definitions being associated with GUI elements. Furthermore, nothing in Carey as referenced by the Examiner has anything to do with examining a runtime version of a screen element of a graphical user interface to detect an ability to process an input device's events and automatically identify the screen element as supporting the input device when input device-handling program code is associated with the screen element. Simply because a GUI element is developed in an object oriented programming language that utilizes classes does not make it obvious to one of ordinary skill in the art to examine the classes to identify methods (i.e., code) that are used to support input devices as claimed by the present invention.

Summary

Neither the teachings nor the nature of the problem solved in either Crutcher or Carey, or the combination thereof, motivate or suggest to one of ordinary skill in the art at the time of the invention to combine the reference teachings in a manner that would make the claimed invention obvious. Furthermore, neither Crutcher nor Carey, nor the combination thereof, teach all of the features of claims 10 and 12-21 as amended. Thus, a *prima facie* case of obviousness has not been established. For at least these reasons, the Applicants respectfully request that the rejections of independent claims 10, 15, 18, and 21 be withdrawn. For at least the same reasons, the Applicants respectfully submit that dependent claims 12-14, 16-17, and 19-20 are patentable over the cited art of record.

Accordingly, a notice of allowance is respectfully requested. Alternatively, the Applicants submit that the claims, in view of the art of record, are in condition for Appeal. If the Examiner has any questions concerning the present amendment, the Examiner is kindly requested to contact the undersigned at (408) 749-6900 x6903. If any additional fee other than the enclosed fees are due in connection with filing this amendment, the Commissioner is authorized to charge to such fees Deposit Account No. 50-0805 (Order No. SUNMP068). A duplicate copy of the transmittal is enclosed for this purpose.

Respectfully submitted,
MARTINE & PENILLA, LLP

[illegible]

Albert S. Penilla, Esq.
Reg. No. 39,487

MARTINE & PENILLA, LLP
710 Lakeway Drive, Suite 170
Sunnyvale, CA 94085
Telephone: (408) 749-6900
Customer Number 32291

COPY OF PAPERS
ORIGINALLY FILED

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the application of

PABLA

Application No. 09/201,644

Filed: November 30, 1998

For: METHOD AND APPARATUS FOR DETECTING DEVICE SUPPORT IN A GRAPHICAL USER INTERFACE

Group Art Unit: 2174

Examiner: Sax, S.

Attorney Docket No. SUNMP068

Date: August 6, 2002

MARKED UP AMENDMENT

MARKED UP CLAIMS

10. (Amended) In a computer system, a method of determining input device support of a screen element of a graphical user interface comprising:

examining a class definition of a screen element of a graphical user interface to detect an ability to process an input device's events, wherein said examining is performed without execution of said class definition; and

automatically identifying said screen element as supporting input device input if said class definition includes a method supporting said input device's input.

15. (Amended) A computer program product comprising a computer usable medium having computer readable program code embodied therein for detecting input device support of a screen element of a graphical user interface comprising:

computer readable program code configured to cause a computer to examine a runtime version of said screen element of a graphical user interface without executing said runtime version of said screen element; and computer readable program code configured to cause a computer to automatically identify said screen element as supporting an input device when input device-handling program code is associated with said screen element.

18. (Amended) A computer comprising:

a display for displaying at least one screen element of a graphical user interface;

at least one input device; and

a detector configured to examine a runtime version of said screen element, without executing said runtime version of said screen element, to automatically identify whether said screen element supports said input device by determining whether input device-handling program code is associated with said screen element.

21. (Amended) In a computer system, a method of detecting input device support of a screen element of a graphical user interface comprising:

automatically examining a runtime version of code associated with a screen element to detect an ability to process events associated with input devices, wherein said automatic examination is performed without executing said runtime version of code associated with said screen element;

automatically determining at runtime whether said screen element delegated processing of said events associated with said input devices to other program code; and

automatically examining said runtime version of code associated with said screen element to detect a declaration of program code that is indicative of support associated with said screen element[s] for a given input device, wherein said automatic examination is performed without executing said runtime version of code associated with said screen element.